Dynamic 3D Maps as Visual Interfaces for Spatio-Temporal Data

Jürgen Döllner Oliver Kersting

Institut für Informatik, University of Münster
Einsteinstr. 62
48149 Münster, Germany
+49 251 83 337 53

{dollner, kerstio}@uni-muenster.de

ABSTRACT

Dynamic 3D maps represent visual interfaces used to present and explore spatial and spatio-temporal data. They provide powerful design capabilities for map contents compared to current map toolkits and general-purpose 3D graphics systems. The underlying object model introduces abstract building blocks which are configured for individual animated, interactive 3D maps. These building blocks do not only include visual primitives but also structural and behavioral primitives: Structural primitives permit to arrange and hierarchically organize 3D map contents, and behavioral primitives define the dynamics and interactivity of 3D maps. The building blocks also support dynamic design of map contents to facilitate visualizing temporal data and phenomena. An embedded scripting language assists to configure 3D maps at run-time as well as to customize and extend 3D maps by the user. Possible applications include interactive, animated cartography, virtual geo-environments, and exploratory, visual interfaces for GIS.

KEYWORDS

Information Visualization, Geographic Visualization, Animated Cartography, Interactive Mapping, Interface Design.

1. INTRODUCTION

Maps are fundamental tools to communicate spatial information. The cartographic abstraction underlying their design has proven to be an effective tool to provide understanding of spatial objects and their relationships. For digital maps, the methods for cartographic abstractions have to be enhanced to take advantage of interactivity, intelligence and varying representation styles – a dynamic map design is required.

The integration of maps in virtual environments offers exciting possibilities to develop new forms of maps and map use that take advantage of the characteristics of virtual environments such as user-map interactivity, user-environment immersion, varying information intensity in the display, and intelligence of display objects [14]. These maps are three-dimensional objects, which are specified by an underlying map terrain model and can be further equipped with 3D objects.

This paper describes the software architecture of a system that implements dynamic 3D maps. The architecture defines an object model that provides abstract building blocks used to construct animated, interactive visual interfaces for GIS applications.

Three categories of building blocks are distinguished. *Visual building blocks* are used to represent geometric objects and to specify the appearance of geometric objects. *Structural building blocks* arrange and hierarchically organize 3D maps and their contents. *Behavioral building blocks* define interactivity and dynamics of 3D maps.

The architecture for dynamic 3D maps supports *dynamic content design*, i.e., the configuration of 3D maps and their contents at run-time. This characteristic facilitates the implementation of varying information intensity and the visualization of temporal data and phenomena.

Technically, an embedded scripting language assists to configure 3D maps at run-time and to customize 3D maps by the user. Multiresolution models for map geometry and map textures guarantee the real-time display of 3D maps.

The presented system for dynamic 3D maps is intended to construct customizable, visual interfaces for spatial data and spatio-temporal data. In particular, it provides powerful design capabilities for map contents compared to current map toolkits and general-purpose 3D graphics systems. Possible applications include interactive, animated cartography, virtual geo-environments, and exploratory, visual interfaces for GIS. Dynamic 3D maps could be deployed in GIS, for example, as embeddable software components.

2. RELATED WORK

Cartographic abstraction as an effective tool to present, explore and manipulate spatial and spatio-temporal information has been applied to virtual environments (MacEachren et al. [14]) in recent years. Virtual geo-environments, which take advantage of the human sensory and cognitive system using VR technology (MacEachren et al. [13]), rely on digital maps as fundamental graphics primitives. The main requirements for these maps are multiresolution and multi-view representation, real-time visualization, interactivity, and high visual quality (Terribilini [19]). In our approach, dynamic 3D maps satisfy these requirements by a hybrid, multiresolution terrain model and a texture-based map design.

8th ACM Symposium on GIS 11/00 Washington, D.C., USA
© 2000 ACM ISBN 1-58113-319-7/00/0011...$5.00

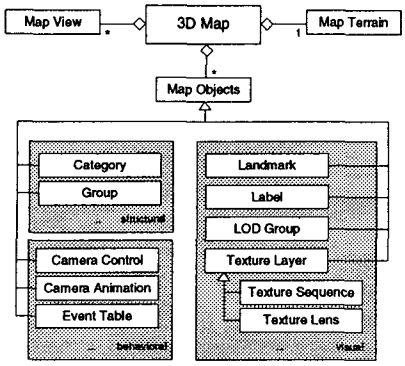

Figure 1. Basic building blocks for dynamic 3D maps.

Davis et al. [5] explore a systematic approach for multiple representations of data in GIS which can be achieved with minimum data redundancy by a comprehensive set of basic operators from computational geometry, spatial analysis, and map generalization. This way, maps are able to cope with different needs of GIS application with respect to resolution, style, and level-of-detail of the visual representation. In our approach, the embedded scripting, which can be used to program application-specific map design in a procedural way, permits the implementation of these concepts.

The design of 3D maps has been investigated by Haeberling [10], who identifies three conceptual aspects: determination of user groups, specification of user needs, and derivation of requirements for map symbolization. We address these aspects by supporting dynamic content design for 3D maps and focusing on mechanisms for customizing and configuring map contents. In particular, GIS often require 3D maps that provide a more schematic than photorealistic visual appearance in order to be informative. Ervin [8] achieves a schematic design by dynamic selection of various map elements such as 3D objects and glyphs. Our building blocks for 3D maps can be used to implement this approach.

The Descartes system (Andrienko et al. [1]), which offers automated knowledge-based visualization and dynamic manipulation, supports exploratory analysis of spatial information based on 2D maps. However, our work concentrates on the system architecture for exploratory 3D maps. It provides the technical mechanisms upon automated, knowledge-based visualization strategies can be implemented.

In the context of computer graphics, several general-purpose systems (e.g., VRML [3] and Java3D [18]) are available that support the implementation of virtual environments. These systems, however, do not concentrate on geo-visualization. Therefore, geo-visualization techniques required by 3D maps such as efficient multiresolution terrain models, texture-based dynamic content design for maps, and dynamic map configuration and customization are difficult to program. GeoVRML (Reddy et al. [17]) attempts to integrate geo-specific visualization capabilities. In our approach, we focus on abstract building blocks for constructing 3D maps. Therefore, map developers are less concerned with learning programming a complex 3D graphics system but can concentrate on extending pre-built 3D map components using a simple scripting language. It is possible to integrate these components as geo-specific extensions to general-purpose 3D graphics systems.

3. ELEMENTS OF 3D MAPS

Interactive 3D interfaces for GIS become increasingly important but are difficult to develop with current visualization and graphics systems. These systems provide general-purpose primitives but neither suitable abstractions nor efficient algorithms for the specific needs of geo visualizations.

This has been the motivation for us to seek for a collection of general-purpose 3D map components. These components should allow developers to construct customized, efficient visual interfaces for spatial and spatio-temporal data.

We propose a high-level object model for dynamic 3D maps that provides components for map terrains, map content design, map dynamics, and map interactivity. A concrete 3D map is represented by a map terrain model, one or more map views, and a collection of map objects. Map objects are either visual, behavioral, or structural. The basic building blocks for 3D maps are shown in Figure 1.

3.1 Map Terrain Model

As a fundamental primitive, 3D maps require a multiresolution terrain model (e.g., multi triangulations [9], progressive meshes [12]) to guarantee interactive frame rates. As a common limitation, these multiresolution models do not integrate multiresolution modeling for texture data. However, textures are fundamental graphics primitives to design map contents, e.g., to visualize thematic data in 3D maps. Furthermore, texturing provides image operations such as filtering and blending, efficiently supported by today's graphics hardware.

In our approach, we apply the *approximation tree* [2], a multiresolution terrain model implementing level-of-detail algorithms for both terrain geometry and terrain textures. It permits the exact representation of terrain morphology by

Figure 2. Landmarks used to depict ways (left). Label and an interactive landmark (center). 3D map with texture layers for shading, topographic, and water flooding information (right).

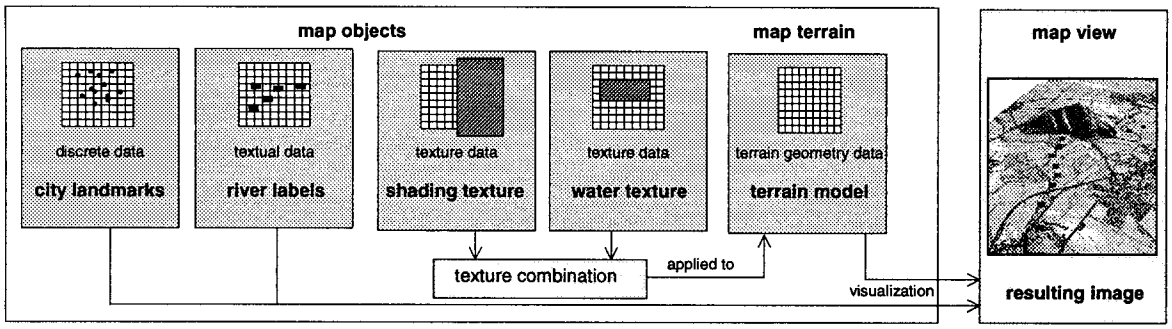

Figure 3. Conceptual view of a concrete 3D map and its visualization.

combining regular grid data with additional TIN data; TINs are used, for example, to refine terrain regions with complex morphology. The complementary *texture tree* [6] implements a multiresolution model for 2D terrain textures. Multiple texture trees can be associated with an approximation tree. This way, a 3D map can be rendered with multiple, possibly overlapping textures. For example, Figure 2 (right) shows a 3D map with one shading texture and two thematic textures.

3.2 Visual Map Objects

Visual map objects represent 2D and 3D geometric objects that are added to the terrain model such as labels, 3D models, and 2D terrain textures. A visual map object is geo-referenced and characterized by graphics attributes (e.g., color, font, size, orientation, shape) used as cartographic variables. Compared to low-level primitives provided by 3D graphics systems, visual map objects exhibit a complex structure and behavior. Visual map objects include:

Landmarks are geo-referenced 2D or 3D symbols and objects used, for example, to mark discrete geo objects and positions, to indicate ways and directions (Elvins et al. [7]) and to represent thematic information (Figure 2 left).

Labels are textual descriptions that are associated with geo positions in the 3D map (Figure 2 center). Usually, labels are located in the view plane and oriented towards the camera direction automatically (billboarding technique).

Texture layers are 2D images superimposed on the terrain model. A texture layer, which may cover the whole or part of the terrain, is positioned due to its geo coordinates. Texture layers may be visually combined in a 3D map view (Figure 2 right).

Level-of-Detail Groups control the complexity of visual map objects. Based on the distance between camera and a reference map point, the LOD group selects an appropriate group element for display.

To create visual map objects, the user can import geometric 3D models or select from a collection of built-in shapes. In addition, scripts are able to automated the construction of visual map objects using the embedded scripting language. For example, visual map objects are frequently used to depict the results of GIS inquiries.

3.3 Structural Map Objects

Structural map objects are used to conceptually arrange map objects and to expose relationships. Structures provide means for users to classify and organize large, individual collections of map objects. In 3D maps, map objects are organized hierarchically; the hierarchies can be manifold. Structural map objects include:

Categories provide classifications for map objects. Associating a map object and a category adds meta information to the map object. For example a thematic texture that represents a forest can be associated with the categories "plants" and "deciduous forest".

Groups arrange map objects in application-specific divisions. groups are containers for map objects; groups can be nested. A map object can belong to more than one group. For example a collection of landmarks and 3D models representing the buildings of a city can be gathered into a group.

3.4 Behavioral Map Objects

Behavioral map objects specify how map objects react to events and state changes. Consequently, behavioral map objects define the interactivity and dynamics of 3D maps and their map objects.

Behavioral map objects are modeled as separate entities to be able to assign map objects an individual behavior independent from the class from which the map objects are instantiated. Therefore, application-specific behavior needs not to be defined by the map object classes but can be specified by associating visual and structural map objects with behavioral map objects.

3.4.1 Event-Callback Tables

The *event-table* represents a generic behavioral map object which associates events and callbacks. Every visual and structural map object can be associated with event-tables.

Events include device events (e.g. mouse clicks or movements), time events, and state changes of map objects (e.g., visibility state of objects). Callbacks represent actions to be executed in the case of events. A callback is implemented as a script which is executed by the embedded Tcl interpreter. Script-based callbacks offer developers and map users a simple way of programming maps at run-time (see Table 1).

Map objects can share event-tables, i.e., behavior can be grouped. For example, a thematic texture, a group of glyphs and a text label react in the same way if they are associated to the same event-table.

3.4.2 Constraints

Constraint map objects are another category of behavioral map objects. For example, the *camera-animation map object* moves and orients the camera along a defined path through the 3D map, reacting to time events. The *camera-control map object* ensures that user defined constraints on camera parameters are satisfied.

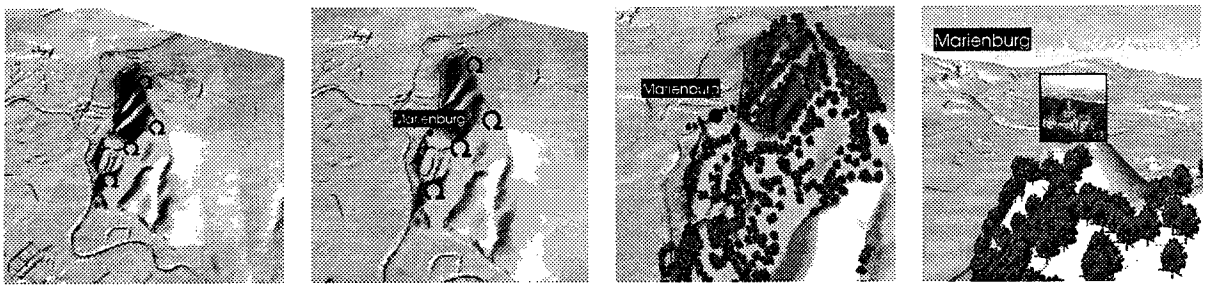

Figure 4. Level-of-detail representation of a geo object (castle) and a geo field (forest land-use).

For example, constraints may specify a maximal camera height or restrict the degrees of freedom during interactive navigation.

3.5 Map Views

Map views visualize 3D maps in a 3D graphics window. Map views are implemented based on a 3D real-time graphics system such as OpenGL [20]. For a given 3D map, the user can create any number of map views and specify independently camera settings. Figure 3 shows the conceptual structure of a concrete 3D map and its visualization in a 3D map view.

3D map views can visualize the same 3D map differently. For example, the main map view could use a perspective projection, while on the contrary an overview map view, which supports indirect navigation and provides orientation, could use an orthographic projection.

4. DYNAMICS OF 3D MAPS

Animated cartography is investigating the potential of dynamic map representations for communicating geo processes and geo phenomena. Results in multimedia cartography show that animated and interactive visualizations provide means for an improved and more intuitive perception of spatial and spatio-temporal data [4]. Spatial data whose shape, size, etc. change over time are represented in 3D maps by creating and reconfiguring map contents or animating map objects. The ability to customize 3D maps at run-time, which is based on the internal interpretative scripting language, supports the development of visualization strategies for spatio-temporal data.

4.1 Level-of-Detail Groups

Thematic information should be visualized according to the user's needs, the screen resolution, and with an appropriate information intensity [13]. Otherwise, a 3D map could be visually overloaded, and its cognitive load might become too high.

In 3D maps, the *LOD group* map object represents thematic data in varying intensity and thus provides means for cartographic generalization. The LOD group contains a collection of distance-dependent, possibly inhomogeneous map objects that describe thematic data at different levels of detail. For each map object contained in an LOD group, an interval specifies the distance range for which the object is visible. The distance intervals of map objects can overlap in order to blend between two levels of detail.

Figure 4 illustrates the LOD representation for geo objects and geo fields. The geo object, a castle, is represented by a box and by a photography. The geo field, a forest land-use data set, is visualized by abstract and concrete glyphs.

4.2 Texture Sequences

A sequence of textures can be used to encode time-varying thematic data. Each texture specifies a key frame for animation. During animation, the textures of two consecutive key frames are blended with appropriate weights. No intermediate texture has to be created (e.g., using image operations) because the textures are combined in screen space. Thus, even high-resolution textures can be directly animated in real-time.

Texture-sequence map objects are specialized texture layers which assign a time stamp to each texture they store. The time used for interpolation can be controlled by GUI components (e.g., scrollbar) or by time events. Figure 5 shows a texture-sequence map object visualizing the flooding state in a landscape.

4.3 Texture Lenses

Texture lens map objects are specialized texture layers which modify the visibility and combination of thematic texture layers. This way, we can focus or reduce information dynamically in a 3D map.

A *luminance texture lens* modifies the brightness of another thematic texture layer. By animating the luminance texture lens, users can be guided through 3D maps (e.g., to explain ways).

A *filter texture lens*, in contrast, restricts the visibility of

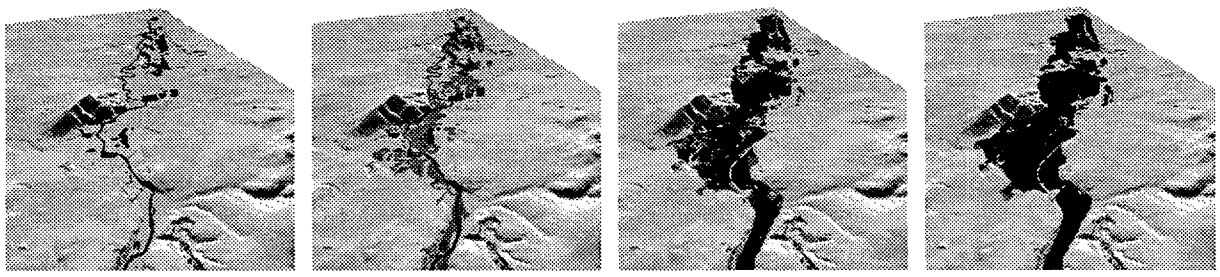

Figure 5. Flooding animation implemented by a texture-sequence map object. The animation results from interpolating between consecutive key-frame textures.

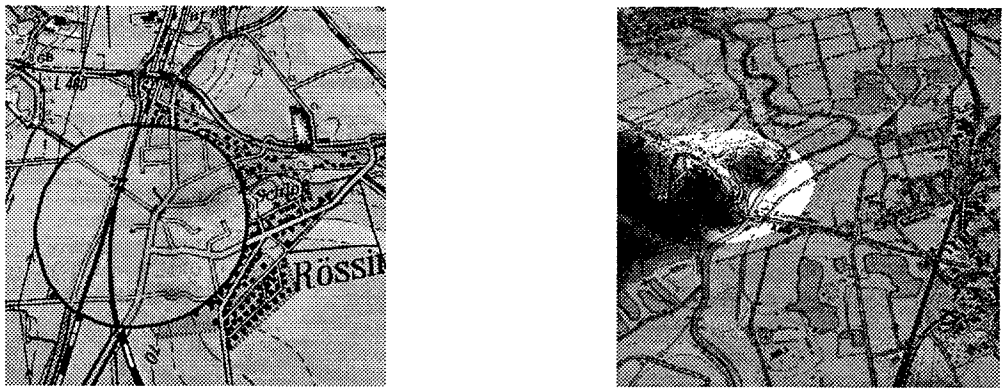

Figure 6. Applications of texture-lens map objects: Information restriction by fading out thematic information (left) and information focussing by hightlighting (right).

another thematic texture layer. It can be used to simplify the visual representation by removing thematic data in the 3D map.

Filter texture lenses allow users to define where information becomes visible. Therefore, the complexity of the map contents can be partially reduced: the 3D map shows those map elements which are important for the current context (e.g., Figure 6 left). Luminance texture lenses allow users to highlight information. This way, for example, the user's attention can be guided (e.g., Figure 6 right).

Table 1. Tcl commands for label map objects.

Operation	Tcl command
construction	`create LABEL myLabel`
destruction	`delete myLabel`
modification	`configure myLabel -color {1 0 0}`
inquiry	`info myLabel -color`

5. SCRIPTING FOR 3D MAPS

The two key characteristics that make 3D maps dynamic and flexible are configuration and customization. To configure 3D maps, it must be possible to create, manipulate, destroy, and associate map objects at run-time. Customization is required to satisfy the very different needs of applications and user requirements.

Table 2. Properties of label map objects.

Property	Domain	Meaning
`-billboard`	true,false	automatic label orientation
`-categories`	{c1 ...cN}	associated categories
`-color`	{r g b a}	label color
`-distanceCall`	tclScript	distance callback
`-enabled`	true,false	enabled flag
`-intersectCall`	tclScript	intersection callback
`-llf`	{x y z}	lower corner of boundingbox
`-offset`	{x y z}	label viewplane offset
`-position`	{x y z}	label position
`-size`	int	label size
`-text`	labeltext	label text
`-urb`	{x y z}	upper corner of boundingbox
`-viewcoords`	true,false	coordinate system
`-visibleCall`	tclScript	visibility callback

In general, programming at class level (e.g., C++) requires high-level programming skills and involves compiling and linking whenever the coding is changed. Hence, it is not a good choice for configuring and customizing 3D maps from a map developers' point of view.

In our approach, we are using an embedded scripting language Tcl [15] to configure and customize 3D maps. The scripting language is intended to be a gluing language [16] to easily connect existing software systems and components into a single application. Since Tcl uses a simple syntax and due to its interpreter nature, it is much easier and faster to develop applications than in traditional system programming languages. A large collection of Tcl plug-ins are available [11] such as database interfaces and constraint solvers. These plug-ins can extend the 3D map system as well.

5.1 Managing Map Objects

The map objects of a 3D map are collected by a project object which takes care of saving and loading map objects. Table 1 illustrates the generic commands used to manage map objects for the case of labels; Table 2 shows the corresponding list of label properties used by these commands.

5.2 Specifying Interactivity of Map Objects

Map objects respond to state changes (e.g., visibility and distance with respect to the camera) and events (e.g., mouse selection, time events) by user-defined callbacks which are specified by Tcl scripts. This way, the user can create individual behavior for each map object.

For example the intersection callback is invoked if the user picks a visual map object. The callback parameters are the 3D intersection point and the name of the picked object. A list of common callbacks for map objects is given in Table 3.

Callbacks are typically implemented as Tcl procedures. The following built-in variables are predefined: `%t` contains the map-

Table 3. Common callbacks for map objects.

Type	Domain	Invocation
visibleCall	true,false	camera changes
distanceCall	[0,+∞)	camera changes
intersectCall	{x y z}	object hit
resultCall	*object dependent*	*object dependent*

object type (e.g., LABEL), %n the map-object name (e.g., myLabel), and %v the value from the callback's domain (e.g., true).

Example 1: The following coding specifies two callbacks for a label map object. The first callback (*visibleCall*) pops up automatically an additional information window if the label is actually visible in the map view. The second callback (*distanceCall*) sets up the description of the information window depending on the distance of the label: if the label is far away, a short description is used, otherwise a more detailed description appears. The example shows how arbitrary, user-specific Tcl procedures customize 3D maps.

```
proc showInfoBox {visible} {
  # show info window
  if {$visible} { wm deiconify .info }
  # hide info window
  else { wm withdraw .info }
}
proc setInfoText {distance} {
  if {$distance > 1000} {
    # use short description
    .info.text configure -text $shortDescription
  } else {
    # use long description
    .info.text configure -text $longDescription
  }
}
# add callbacks to myLabel
configure myLabel \
    -visibleCall {showInfoBox %v} \
    -distanceCall {setInfoText %v}
```

Example 2: In the following coding, an interactive texture lens is specified. If the user clicks on the map terrain, a 3D intersection point is calculated and passed to the intersection callback. The intersection callback sets the lens center to that point. Therefore, the user can move the lens interactively on the terrain. In addition, a slider is added to the user interface which controls the size property, i.e., the radius of the texture lens.

```
# create lens map object
create LENS myLens -mixer "lens.bmp" \
  -texture1 "complexthematic.bmp" \
  -texture2 "streets.bmp" -size 1000

# bind lens center to intersection point
configure myTerrain -intersectCall \
    {configure myLens -center %v}

# create slider controlling lens size
scale .mySlider -from 1 -to 1000 -resolution 5 \
    -command {configure myLens -size}
```

In our experience, the hybrid approach consisting of a system programming language used to implement 3D map object classes and a scripting language used to configure and customize 3D map objects represents a powerful combination. Although most commands are triggered through the scripting language, C++ objects finally execute the time-critical operations. Therefore, no significant performance is lost.

6. CONCLUSIONS

The presented system for dynamic 3D maps offers abstract building blocks for constructing visual interfaces for spatial data and spatio-temporal data. Abstract building blocks are essential to encapsulate complex algorithms necessary to cope with real-time requirements and the specific needs of dynamic geo-visualization techniques. In particular, our approach provides map objects that support animated, interactive information display with varying information intensity to adjust the map contents according to the screen resolution and the users' needs. Abstract building blocks in combination with an embedded scripting language simplify the rapid development of customizable, visual interfaces for any kind of GIS application using maps as fundamental tools.

REFERENCES

[1] Andrienko, G., and N. Andrienko, "Interactive Maps for Visual Data Exploration," *International Journal Geographic Information Science*, 13(4), 355-374 (1999).

[2] Baumann, K., J. Döllner, K. Hinrichs, and O. Kersting, "A Hybrid, Hierarchical Data Structure for Real-Time Terrain Visualization," *Proceedings Computer Graphics International '99*, 85-92 (1999).

[3] Carey, R., and G. Bell, *The Annotated Vrml 2.0 Reference Manual*, Addison-Wesley (1997).

[4] Cartwright, W., M.P. Peterson, and G. Gartner, *Multimedia Cartography*, Springer Verlag (1999).

[5] Davis Jr., R., and A. Laender, "Multiple Representations in GIS: Materialization Through Map Generalization Geometric, and Spatial Analysis Operations," *Proceedings of ACM GIS '99*, 60-65 (1999).

[6] Döllner, J., K. Baumann, and K. Hinrichs, "Texturing Techniques for Terrain Visualization," *Proceedings IEEE Visualization 2000*, in print (2000).

[7] Elvins, T., D. Nadeau, and D. Kirsh, "Worldlets - 3D Thumbnails for Wayfinding in Virtual Environments," *Proceedings of UIST 97*, 21-30 (1997).

[8] Ervin, S., "Landscape Visualization with Emaps. *IEEE Computer Graphics & Applications*, 28-33 (1993).

[9] De Floriani, L., P. Magillo, and E. Puppo, "Efficient Implementation of Multi-Triangulations," *Proceedings IEEE Visualization '98*, 43-50 (1998).

[10] Haeberling, C., "Symbolization in topographic 3D maps: Conceptual Aspects for User-Oriented Design, *19th International Cartographic Conference*, 1037-1044 (1999).

[11] Harrison, M., *Tcl/Tk Tools*, O'Reilly & Associates, Chapter 11 (1997).

[12] Hoppe, H., "Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering," *Proceedings Visualization '98*, 35-42 (1998).

[13] MacEachren, A.M., R. Edsall, D. Haug, R. Baxter, G. Otto, R. Masters, S. Fuhrmann, and L. Qian, "Exploring the Potential of Virtual Environments for Geographic Visualization," *Association of American Geographers* (1999).

[14] MacEachren, A.M., M.-J. Kraak, and E. Verbree, "Cartographic issues in the design and application of geospatial virtual environments," *19th International Cartographic Conference*, 657-665 (1999).

[15] Ousterhout, J., *Tcl/Tk*, Addison-Wesley (1994).

[16] Ousterhout, J., "Scripting: Higher Level Programming for the 21st Century," *IEEE Computer*, 31(3), 23-30 (1998).

[17] Reddy, M., L. Iverson, and Y. Leclerc, "Under the Hood of GeoVRML 1.0," *Proceedings of The Fifth Web3D/VRML Symposium*, 23-28 (2000).

[18] Sowizral, H., K. Rushforth, and M. Deering, *The Java 3D API Specification*, Addison Wesley (1997).

[19] Terribilini, A., "Maps in transition: development of interactive vector-based topographic 3D-maps," *19th International Cartographic Conference*, 993-1001 (1999).

[20] Woo, M., J. Neider, T. Davis, and D. Shreiner, *OpenGL Programming Guide – 3rd edition*, Addison-Wesley (1999).